2.6 WEAPONS

The weapon is one of the eight system types in Suppressor. Its role is to inflict damage to an element of another player. Weapons possess a resource known as ordnance which travels (i.e. flies out) towards the intended target. There are two major ways in which Suppressor weapon systems are categorized. This first is whether the ordnance flyout is implicit or disaggregated (i.e. explicit). This distinction is described below. The second category is whether the weapon is guided (controlled) or unguided (uncontrolled). This distinction is illustrated in functional element design approach 2.6.2.

As mentioned above, there are two ways to model ordnance flyouts, implicit and disaggregated (i.e. explicit). To model an implicit flyout, the database developer specifies an ORDNANCE type of resource under the WEAPON system identification in the shooting player's PLAYER-STRUCTURE. When using an implicit flyout, a calculation is performed at weapon firing time which determines, using the known future movement path of the target and the Weapon Speed vs. Time capability of the ordnance, whether the ordnance can intercept the target within its maximum flyout time. This maximum flyout time as well as the Weapon Speed capability are user inputs. If the intercept can occur, then an event is scheduled at the calculated intercept time. This event performs the random number draw to determine the lethality of the shot. If the target is too far away and the intercept cannot occur, then that fact is reported to the user in the output file at the time of the shot.

Events occurring during the flyout can alter the initial intercept determination. If the target performs a reactive maneuver while the ordnance is in flight, another calculation is done to determine if an intercept can occur within the remaining flyout time. Therefore, an intercept determined to be possible at firing time could become unattainable and vice versa. Finally, it should be noted that an implicitly fired ordnance is not a trackable entity in Suppressor. This means that it cannot be detected by sensors nor can its trajectory be accurately determined.

To model a disaggregated or explicit flyout, a FUTURE-PLAYER type of resource is specified under the WEAPON system in the shooting player's PLAYER-STRUCTURE. The ordnance is an actual Suppressor player created at the time of firing. This player is structured like any other player, including having a PLAYER-STRUCTURE entry in the TDB. Because it will move, it will contain a mover system. To model its lethality, the player will contain its own weapon system and tactics describing the engagement and firing (i.e. detonation) conditions. It may own sensors (i.e. seekers) and/or communications systems. In the SDB, a template for the disaggregated players is placed subordinate to the player which will fire it. The template describes details which all instances of the disaggregated player will share. Because a disaggregated ordnance is a separate player, it has the potential to be detected and engaged by other players and its trajectory viewed as any other moving player in Suppressor.

Finally, a weapon system can be defined to cause the shooting location to self-destruct upon firing of its ordnance. Such a weapon might be a warhead on a disaggregated missile which causes the entire missile player to die as the result of its detonation. Either the entry SELF-DESTRUCTION or NO-SELF-DESTRUCT will appear in the weapon system capability's WPN-CHARACTERISTICS data item

2.6.1 Functional Element Design Requirements

The design requirements for the weapons functional element are:

a. Provide the capability for a user to create a player with a weapon that can be either implicitly or explicitly fired at a target. For an implicit flyout, the model will compute the weapon-target intercept based on the weapon-target geometry and on a user-specified weapon speed profile. For an explicit flyout, the model will implement the creation of the disaggregated weapon at the appropriate time within the simulation and use reactive movement instructions to determine its flight path as it attempts to intercept the target. The disaggregated weapon will be modeled as a separate player with user-specified attributes and characteristics.

b. Determine whether or not an implicit-flyout weapon can intercept its target based on relative target/weapon geometry and on user-specified weapon speed or movement path. If it cannot intercept the target, the weapon will abort and disappear from the scenario. If it can intercept the target, the model will calculate the resultant damage based on the relative position of the shooter and the vulnerability of the target defined in terms of a probability of kill table for the weapon/target combination.

c. Provide the capability for the user to define the speed profile of ordnance as it implicitly moves out from the shooter to the target. The speed of the launcher at the time of launch will be added to these speeds, if the user so chooses. The model will use this data to calculate the approximate time and position of the intercept for implicit flyouts. The model will also use this speed profile to calculate the maximum range and maximum flyout time for the weapon and to determine if the intercept is possible or if an abort event should be scheduled.

d. Provide the capability for a user to specify characteristics for each implicit-flyout weapon type. The user options will include the following instructions:

    1. The time and position of the possible intercept will be computed by considering the x-y and altitude differences between the weapon location and the target location.

    2. The time of the geometry calculations for the p(k) table lookup can be either at launch or at intercept. If the p(k) table depends on relative weapon-target geometry, the positions at either the time of launch or the time of intercept will be specified by the user.

    3. The weapon's mode of control after launch can be either controlled or uncontrolled. Controlled weapons are guided by the shooter after launch; the launcher has no control over uncontrolled weapons after launch and does not know when intercept occurs.

    4. The option for self-destruction will be provided such that both the weapon and its location are destroyed at intercept.

e. The model will provide the capability for the user to define two time delays for a weapon: the time delay between the decision to fire and the firing of the first round and the time delay between firing two rounds in a salvo. The time delay

between the decision to fire and the start of movement will be used to compute a total flyout time. If an abort event is scheduled, the model will use the total flyout time to compute the time of the abort event. The time delay between firing two rounds or a salvo to will be used to schedule subsequent weapon firing events for a weapon.

f. Provide the capability for a user to define the probabilities of kill for a given type of implicit-flyout weapon ordnance against a given type of target by creating a lookup table. This table is described in Section 2.4 for the Vulnerability FE.

g. The user can specify a maximum number of targets that can be engaged in parallel by a weapon type.

h. A weapon reload option will be provided. The user can specify the number of extra rounds available, the amount of time needed to reload a given quantity of rounds, the threshold number of rounds that causes reloading to occur, and the quantity of rounds to be reloaded.

i. The model will prevent a weapon from exceeding any user-defined azimuth and elevation slew limits for its weapon type.

2.6.2 Functional Element Design Approach

Design Element 6-1: Guided Weapons

Simply put, a controlled (guided) weapon system in Suppressor is one in which the ordnance is "linked" to the original shooting location. This basic effect of this linkage is to abort an ongoing flyout and self-destruct the ordnance should the shooting location be killed. Any scheduled intercept events for the ordnance will be canceled.

Another aspect of controlled weapons occurs when the shooting player owns a sensor with tracking capability. The tracking sensor can be associated with the weapon system by including the relationship in the LINKAGES section of the PLAYER-STRUCTURE and by using a WITH-TRACKER phrase in the engagement start RESOURCE-ALLOCATION tactics. If, during an ordnance flyout, the linked tracker goes into coast mode for a time period exceeding its MAX-COAST-TIME input, the flyout will be aborted and the ordnance will self-destruct.

In an implicit flyout, no communication occurs between the shooter and the ordnance because the ordnance is not a player. Therefore, the concept of actual guidance is not modeled. However, in an explicit flyout the ordnance is an actual player. Therefore, if communications equipment is included on that player and placed on a compatible communications net with the shooter, explicit guidance can occur. The shooter will pass on to the ordnance the updated target perception data gained from its tracking sensor.

Design Element 6-2: Unguided Weapons

In contrast to a guided weapon which is linked to the original shooting location, an unguided weapon once fired or launched is completely independent of the shooter. This option is intended to represent ballistic weapons such as bullets or gravity bombs. As for guided weapons, the weapon speed and range is determined by the WPN-SPD-

CAPABILITY data item in the TDB. If an unguided weapon has sufficient range to reach the target, it will always intercept.

2.6.3 Functional Element Software Design

The software modules related to weapons modeling are listed in Table 2.6-1. Routine AILHIT schedules the weapon intercept event for both guided and unguided weapons and uses the following logic:

```
*begin logic to schedule weapon hits:
   *write out situation data;
   *when target cluster exists:
      *meld wpn/tgt buffer into list under cluster;
   *end of test for target cluster.
   *look up pointer to input event and snr/wpn/tgt buffer;
   *when shooter must control ordnance to intercept:
      *invoke logic to find buffer corresponding to this event;
      *invoke logic to schedule intercept/abort for shooter;
      *store pointer to event in buffer and firing event;
      *store id of weapon and interaction key of shooter;
   *otherwise, uncontrolled ordnance intercept:
      *invoke logic to schedule environmental intercept/abort;
      *when sensor/weapon buffer exists:
         *invoke logic to drop buffer from list;
         *invoke logic to delete buffer from list;
      *end of buffer exists.
   *end of test for controlled or uncontrolled.
   *store pointer to weapon firing event;
*end of logic for AILHIT.
```

TABLE 2.6-1. Weapons-related Software Modules.

Modules	Data Block (number)	Data Items
AILHIT DELSWT OBSLOK SIMPHY WPNABO WPNARR WPNFYR WPNHIT	System (4) Weapon Status (51) Weapon Data Base (191)	WPN-CHARACTERISTICS

The design for the code which aborts a flyout when a player dies is found in routine DELSWT:

```
*when cluster associated with a flyout:
   *search for event buffer under shooter's engagements;
   *drop buffer from shooter's engagement list;
   *delete buffer from shooter's weapon list;
*end of test for flyout.
*reset self-destruct flag if necessary;
*loop, while events left to be deleted:
   *when there is an associated event:
      *when it is not an abort event:
         *when target cluster is still alive:
            *delete entry from weapon engaging list;
         *end of test for target cluster.
```

```
            *reset situation code if firing event;
          *otherwise, check type of abort:
            *reset situation code if bad launch;
          *end of test for type of launch.
          *write out "abort" or "bad launch" situation;
          *invoke logic to cancel event;
          *invoke logic to delete buffer from list;
       *otherwise, associated flyout:
          *schedule a forced self-destruction for flyout;
          *invoke logic to delete buffer from list;
       *end of test for associated event.
    *end of loop for events.
    *when self-destruction flag is set:
       *schedule an immediate forced self-destruction;
    *end of test for self-destruction.
```

The design for the code which aborts the flyout when the tracking sensor has been coasting longer than the threshold input value is found in routine OBSLOK:

```
   *when in coast mode and maximum coast time exceeded:
      *loop, while sensor/weapon/target buffers left:
         *when at least one weapon event buffer:
            *loop, while other tracking sensors in engagement:
               *when at least one tracker exists which is not coasting
               * beyond max-coast-time:
            *end of loop for other tracking sensors in engagement.
         *end of test for one weapon event buffer.
         *when no other tracker found:
            *loop, while weapon event buffers:
               *when this is a disaggregated flyout:
                  *invoke logic to destroy the missile;
                  *invoke logic to drop buffer from list;
                  *invoke logic to delete buffer from list;
               *but, when this is not an abort event:
                  *when ordnance is controlled to intercept:
                     *invoke logic to schedule weapon abort;
                     *store pointer to new event;
                  *end of test for controlled intercept.
               *end of test for disaggregated or implicit flyout.
            *end of loop for buffers.
         *end of test for no other tracker.
      *end of loop for entries on weapon/target list.
    *end of test for in coast mode.
```

2.6.4 Assumptions and Limitations

Guidance for controlled, implicit weapons is assumed to be perfect.

2.6.5 Known Problems or Anomalies

None.